REMARKS

This is a full and timely response to the outstanding final Office Action mailed June 21, 2005. Reconsideration and allowance of the application and pending claims 1, 2, and 4-22 are respectfully requested.

I. Claim Rejections - 35 U.S.C. § 103(a)

A. Rejection of Claims 1-2 and 4-22

Claims 1-2 and 4-22 have been rejected under 35 U.S.C. § 103(a) as allegedly unpatentable over *Jackson* ("*Jackson,*" U.S. Pub. No. 2002/0169769) in view of *Ito et al.* ("*Ito,*" U.S. Pub. No. 2002/0016792). Applicants respectfully submit that the rejection has been rendered moot, as described further below.

B. Discussion of the Rejection

Jackson has been applied as the primary reference to claims 1-22. *Ito* has been applied as a secondary reference to claims 1-22. Declarations from the inventors under 37 C.F.R. § 1.131 and documentary evidence corresponding to Exhibits A-D accompanying the declarations have been submitted along with the present Response that identifies that the claimed invention was reduced to practice prior May 12, 2001 and December 20, 2000 filing dates of the *Jackson* and *Ito* publications, respectively. Exhibit "A" is a copy of pages 1-9 of a low level design document entitled, "RapidLogin(Persistent Storage)," which includes details of how to implement an embodiment of the claimed invention corresponding to the Rapid Login feature as part of a software package (ITRC 2.0 Release). This low level design document occurred prior to the earliest critical date of December 20, 2000. Exhibit "B" is a copy of an internal memo that summarizes test results of the software release (ITRC 2.0 Release) that includes an embodiment of the claimed invention corresponding to the Rapid Login feature, and that presents approval by the Program Office and Core release team members to deploy the software. Dates have been redacted in the document. The internal memo provides evidence that an embodiment of the claimed invention was tested and achieved satisfactory performance prior to the earliest critical date of December 20, 2000. Exhibit "C" is a copy of pages 1-2 of a Test Plan for the software release (ITRC 2.0 Release) that includes an explanation of an embodiment of the claimed invention corresponding to the Rapid Login feature. Dates have been redacted in the document. This Test Plan document occurred prior to the earliest critical date of December 20, 2000. Exhibit "D" is a copy of a document entitled, "Rapid Login - Performance Improvement for ITRC Login – Persistent Storage of Permissions," which is a project document that includes details of an embodiment of the claimed invention corresponding to the Rapid Login feature. Dates have been redacted in the document. This document occurred prior to the earliest critical date of December 20, 2000. Thus, all of these Exhibits provide evidence of an embodiment of the claimed invention that has been reduced to practice, and all of these Exhibits have creation dates that predate the May 12, 2001 and December 20, 2000 filing dates (which accordingly provides evidence of an embodiment of the claimed invention that has been reduced to practice before the critical dates).

The final Office Action asserts the following on page 8, section 32:

> The evidence submitted is insufficient to establish a conception of the invention prior to the effective date of the Jackson reference (4/12/2001). While conception is the mental part of the inventive act, it must be capable of proof, such as by demonstrative evidence or by a complete disclosure to another. Conception is more than a vague idea of how to solve a problem. The requisite means themselves and their interaction must also be comprehended. See *Mergenthaler v. Scudder*, 1897 C.D. 724, 81 O.G. 1417 (D.C. Cir. 1897).
> The Exhibits (A-D) draw inference from the cited Rapid Login application that the features of the invention are present in particular embodiments presented in exhibits. There is no proof that the cited exhibits correspond with the claimed elements of the invention. There is not a clear and definite idea of the complete and operative invention which is not sufficient for enablement.

Applicants respectfully disagree. All of the exhibits collectively provide an enabling disclosure of the embodiments of the invention and further provide evidence of conception and reduction to practice. For example, independent claim 1 is reproduced below with an explanation o f s ections from E xhibit D corresponding to each claim element in italics. It is noted that similar support may be applied to independent claim 12.

1. A program for caching an entitlement set, the program being stored as a computer readable medium, the entitlement set designating services and products a user is entitled to access in a network, the program comprising:

<u>(a) logic configured to receive a login request from the user;</u>
Login requests are generally understood by those having ordinary skill in the art, although the functionality of logging into a system is generically referenced in the Purpose section of Exhibit D (i.e., "To reduce worst case login time by eliminating the need... ") as well as under the Benefits (FCC) section ("Login").

<u>(b) logic configured to determine whether a dirty buffer indicating a triggering event related to the user exists, the dirty buffer having been created after a triggering event;</u>
*General reference is made to the following sections of Exhibit D to support this claim element: **Multiple Instances, Behavior of modify_datetime stamp, Call Set and Clear Dirty Permissions Buffer.** Also refer to sections **Manage Dirty Buffer: Set, SetAll, GetClear Permissions Flags.** For example, under the Multiple Instances section, the disclosure asserts that "[I]f multiple instances of the same user add different entities. The copies of the permissions in session would be different. This could be handled by saving the modify date/timestamp and checking at put. If this fails, a forceful write occurs to the dirty buffer marking the entity needs to be replaced in permissions." In other words, the dirty buffer is created and indicates a triggering event relating to the user. Further support is found in the Call Set and Clear dirty Permissions Buffer, which describes events that lead to the set-up of the dirty buffer. A determination of whether a buffer exists can be found under the SetDirty bullet point of the section Manage Dirty Buffer (e.g., "if the buffer exists... ").*

<u>(c) logic configured to read a preexisting entitlement set from a memory element if the dirty buffer does not exist, the preexisting entitlement set indicating a first scope of access to the network; and</u>
*General reference is made to the following sections of Exhibit D to support this claim element:**Manage Dirty Buffer: Set, SetAll, GetClear Permissions Flags**. For example, with regard to the preexisting entitlement sets, the second line in the second paragraph under the Manage Dirty Buffer section provides: "If the buffer is empty, the stored preferences are current." That is, the stored preferences are equated to the preexisting entitlement sets. If the buffer is empty, this means that the preferences are current, and thus there is no need to recalculate (a first scope of access).*

<u>(d) logic configured to calculate a new entitlement set if the dirty buffer does exist, the new entitlement set indicating a second scope of access to the network; and</u>

General reference is made to the following sections of Exhibit D to support this claim element: ***Manage Dirty Buffer: Set, SetAll, GetClear Permissions Flags****. Under the section Manage Dirty Buffer, the following is asserted: If the buffer exists, search for EntityID in list. If in list return if modes are equal, otherwise set mode to replace. If not in list append to list. Write current buffer for UserID." That is, the calculation has to be performed if the dirty buffer exists, since this means a triggering event has occurred. Some of the processes are further described under the Call Set and Clear Dirty Permissions Buffer section. Thus, the second scope of access is where calculations must occur.*

(e) logic configured to allow the user a third scope of access to the network, the third scope of access being the first scope of access or the second scope of access.
General reference is made to the following sections of Exhibit D to support this claim element: ***Call Set and Clear Dirty Permissions Buffer, Manage Dirty Buffer: Set, SetAll, GetClear Permissions Flags****. This claim element is simply a natural extension of the first two scopes of access described above.*

In view of these declarations and supporting evidence, Applicants respectfully submit that the rejection to claims 1-22 is moot since neither *Jackson* nor *Ito* is valid anticipatory references. Therefore, Applicants respectfully request that the rejections be withdrawn as applied to claims 1-22.

CONCLUSION

Applicants respectfully submit that Applicants' pending claims are in condition for allowance. Favorable reconsideration and allowance of the present application and all pending claims are hereby courteously requested. If, in the opinion of the Examiner, a telephonic conference would expedite the examination of this matter, the Examiner is invited to call the undersigned attorney at (770) 933-9500.

Respectfully submitted,

David Rodack
Registration No. 47,034

EXHIBIT A

1 Introduction

This document describes low level design for RapidLogin (persistent storage) project, released in ITRC E.2.0.

2 Scope

The scope of this document is limited only to the following:
- Interfaces between service layer and 'cis' process related with RapidLogin (persistent storage) E.2.0 release (i.e. CI stack).
(Note: 'cis' is CIS FCD process and 'cixsmainp' is CIDB server process)
- Interface between 'cis' and 'cixsmainp' process related with RapidLogin (persistent storage) E.2.0 release.
- The data flow and algorithms are explained for 'cis' and 'cixsmainp' processes.
- Different events, invoking these scenarios are not addressed here.
- mapping of functions (i.e. TOBJ and TVAL) across FCC/FCD and FCD/CIDB is out of the scope of this document.

3 References

- Rapid Login (persistent storage) High Level Design document (explains concept and philosophy behind RapidLogin feature)
- TOBJ tutorial (explains TOBJ (text object) protocol and structures used by it)

4 Interface

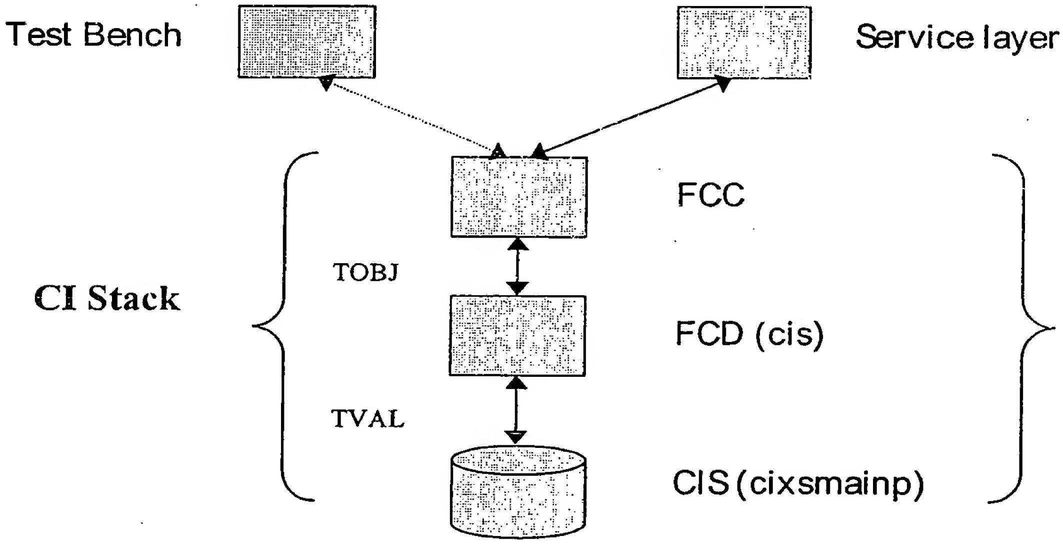


Figure 1 - Control Flow

As shown in Figure 1 - Control Flow whenever a request comes from service layer, it is passed from following layers in the same order, FCC, FCD and CIS.

Each of these 3 layers is explained in separate sections with their interface with their respective layers.

There are 2 main interfaces between service layer and rest of the CI stack, namely a) PutPermissions and b) GetPermissions, regards to RapidLogin project. The actions taken at each of these 3 layers for both the interfaces are also explained.

5 Data Structures

Note: The size of userid, entityid and modify_datetime fields are as per existing fields in other CIDB tables.
Pl check http://nandi.india.hp.com:8000/projects/CIDBHA/WWCIDB/cidb.html - CUSTOMER for more information.

5.1 System Permissions structure (TOBJ format) (ref Permission document)

```
<systems-perms-info> ::= VECTOR { < system-perms-info > }
  <system-perms-info>  ::= TABLE {
    STRING              "system_handle" (system handle or warranty product/serial
                              or e-order ID/item)
    <perms>             "system_perms"
  }

  <perms>          ::= VECTOR { STRING }
```

5.2 EX_PERMISSIONS_2829

```
struct EX_PERMISSIONS_2829 {

   struct EX_CUSTOMER_PERMISSION_2829 {
      char    entityid[51];
      short   sequence;
      char    permission[1001];
      char    modify_datetime[21];
   } ex_customer_permission_2829;

   struct EX_PERMISSIONS_2829  *next;

};
```

5.3 IM_PERMISSIONS_2831

```
struct IM_PERMISSIONS_2831 {

   struct IM_CUSTOMER_PERMISSION_2831 {
      char    userid[21];
      char    entityid[51];
      short   sequence;
      char    permission[1001];
      char    modify_datetime[21];
   } im_customer_permission_2831;

   struct IM_PERMISSIONS_2831  *next;

};
```

Note: The above structures, namely EX_PERMISSIONS_2829 and IM_PERMISSIONS_2831 are generated by OSCAR, an automated tool to create CIDB interface routines.

5.4 CUSTOMER_PERMISSIO table (ref http://nandi.india.hp.com:8000/projects/CIDBHA/WWCIDB/cidb.html - CUSTOMER_PERMISSIO)

6 FCC

The FCC layer is the only layer, from CI stack, seen by service layer. It's the first layer in CI stack.

The requests received by FCC can be one of the followings:

6.1 PutPermissions

Function

This routine acts as an interface between service layer and CI stack. Whenever, service layer issues request for PutPermissions, service layer calls this function from FCC.

Input Parameters

Parameter	DataType	Remarks/Use
userid	char[20]	http://nandi.india.hp.com:8000/projects/CIDBHA/WWCIDB/cidb.html - CUSTOMER
modify_datetime	TOBJ string	YYYYMMDDHHMMSS000000 format (20 characters)
vSystemsPermsInfo	TOBJ vector	in the form of SystemsPermsInfo (refer 5.1)

Output Parameters

Parameter	DataType	Remarks
reply	Perl hashed-array	It contains modify_datetime in YYYYMMDDHHMMSS000000 format (20 characters), received from FCD layer (cis process).

Logic (refer code 13.1.1)

Note:
- *userid* must be a scalar string
- *modify_datetime* must be a TOBJ string
- *vSystemsPermsInfo* must be a TOBJ vector.

This routine reads 3 input parameters as *userid*, *modify_datetime* and *vSystemsPermsInfo* variables.

Then it creates a new TOBJ table (*tobj*), with a size of tobj variable and 3 additional entries for userid, modify_datetime and routing to 'cis' FCD process (used internally by TOBJ library).

After this, it adds *userid* variable as a TOBJ string (by converting scalar string using Smake routine) as 'userid' value in *tobj* table. It adds *modify_datetime* as it is (as its already in TOBJ string format) as 'modify_datetime' value in *tboj* table.

At last it adds, *vSystemsPermsInfo* table of permissions as 'systems-perms-info' value in *tobj* table. This table (i.e. *tobj*) is then passed to TOBJ library with appropriate opcode (i.e., *PutPermissions_opcode*, see /opt/support/fcc/cis/Xaction.pl) value.

The *reply* returned from TOBJ library, is then returned to the service layer as it is.

6.2 GetPermissions

Function

It acts as an interface between service layer and CI stack. Whenever, service layer issues request for GetPermissions, service layer calls this function from FCC.

Input Parameters

Parameter	DataType	Remarks
userid	char[20]	

Output Parameters

Value	DataType	Remarks
reply	Perl hashed-array	It contains modify_datetime in YYYYMMDDHHMMSS000000 format (20 characters), and in SystemsPermsInfo format (ref 5.1), received from FCD layer (cis process).

Logic (refer code 13.1.2)

The *userid* must be a scalar string.

The input parameter *userid* is taken in the args hashed-array as 'userid' value. It is then passed to TOBJ library with appropriate opcode (i.e., *GetPermissions_opcode* see /opt/support/fcc/cis/Xaction.pl) value.

The *reply* returned from TOBJ library, is then returned to the service layer as it is.

7 FCD

7.1 fcdPutPermissions

Function

This function gives a wrapper API for low-level database routine to update CUSTOMER_PERMISSIO table.

When FCC layer sends a request to TOBJ library for PutPermissions function, this routine is called from FCD layer, i.e. 'cis' process (libxaction.sl.s700.10).

Input Parameters

Parameter	DataType	Remarks
req_hdr	TOBJ	
nreq_args	int	
req_args	TOBJ table	This array contains data needed to update record in CUSTOMER_PERMISSIO table
num_dbs	int	
dbs	db_t array	
num_selected_dbs	int	
selected_dbs	int *	
rep_hdr	TOBJ *	
nrep_args	int *	
rep_args	TOBJ **	

Note:
- This is a standard signature for all fcd APIs. Only *req_args* parameter is useful for this project. Rest of the parameters can be ignored.
- Last 3 parameters acts as output parameters.

Output Parameters
See Note above.

Logic (refer code 13.2.1)

This function executes following steps in the same order:
- clears TOBJ error stack.
- retrieves 'userid' and 'modify_datetime' from *req_args* parameter (it's a value-pair array)
- calls tobj_to_i_perms_2831(12.1) routine to convert 'systems-perms-info' TOBJ-structure(5.1) into IM_PERMISSIONS_2831 C-structure (5.3).
- call cisPutPermissions (8.1)
- convert the reply in TOBJ format, if cisPutPermissions is successful.
- prepare reply header, based on return value from cisPutPermissions.

7.2 fcdGetPermissions

Function
This function gives a wrapper API for low-level database routine to retrieve data from CUSTOMER_PERMISSIO table for a given userid.

When FCC layer sends a request to TOBJ library for GetPermissions function, this routine is called from FCD layer, i.e. 'cis' process (libxaction.sl.s700.10).

Input Parameters

Parameter	DataType	Remarks
req_hdr	TOBJ	
nreq_args	int	
req_args	TOBJ table	This array contains data needed to retrieve data from CUSTOMER_PERMISSIO table, i.e. userid.

num_dbs	int	
dbs	db_t array	
num_selected_dbs	int	
selected_dbs	int *	
rep_hdr	TOBJ *	
nrep_args	int *	
rep_args	TOBJ **	

Note:
- This is a standard signature for all fcd APIs. Only req_args parameter is useful for this project. Rest of the parameters can be ignored.
- Last 3 parameters acts as output parameters.

Output Parameters
See Input Parameters' note.

Logic (refer code 13.2.2)

This function executes following steps in the same order:
- clears TOBJ error stack.
- retrieves 'userid' from *req_args* parameter (it's a value-pair array)
- calls cisGetPermissions (8.2)
- if cisGetPermissions is successful, converts the returned permission structure from C to TOBJ format, using e_perms_2829_to_tobj function (12.2).
- prepares reply header, based on return value from cisGetPermissions.

8 CISapi

8.1 cisPutPermissions

Function
This function is called from fcdPutPermissions function.

Input Parameters

Parameter	DataType	Remarks
cidb	char *	
userid	char *	
im_perms_2831	struct * IM_PERMISSIONS_2831 (5.3)	This array contains data needed to update record in CUSTOMER_PERMISSIO table.
ex_cust_perm_2831	struct * EX_CUSTOMER_PERMISSION_2831	

Note:
- Here, ex_cust_perm_2831 acts as an output parameter.

Output Parameters
See Input Parameters' note.

Logic (refer code 13.3.1)
This function executes following steps in the same order:

- clears TOBJ error stack.
- calls update_customer_permissions function (9.1)from libcixcapi.sl library. (libcixcapi.sl is a library generated by OSCAR generated code. No, manual modification is required to generate this library.)
- returns result of update_customer_permissions function call to fcdPutPermissions function, with appropriate error message incase of error.

8.2 cisGetPermissions

Function

This function is called from fcdGetPermissions function.

Input Parameters

Parameter	DataType	Remarks
cidb	char *	
userid	char *	
count	int *	
ex_perms_2829	struct * EX_PERMISSIONS_2829	refer 5.2

Note:
- Here, ex_perms_2829 acts as an output parameter.

Output Parameters

See Input Parameters' note.

Logic (refer code 13.3.2)

This function executes following steps in the same order:
- clears TOBJ error stack.
- calls read_customer_permissions function (9.2) from libcixcapi.sl library. (libcixcapi.sl is a library generated by OSCAR generated code. No, manual modification is required to generate this library.)
- if update_customer_permissions function fails, it returns error code to fcdPutPermissions function, with appropriate error message.
- it calls BuildCustPermRtnList function to make a return buffer of permissions from a linked list of permissions.
- returns appropriate value back to the calling routine, i.e. fcdGetPermissions.

9 CIDB

9.1 update_customer_permissions_p

Function

This function is invoked when a request comes to 'cixsmainp' process, from 'cis' process to update records in the CUSTOMER_PERMISSIO table.

Input Parameters

Parameter	DataType	Remarks
im_permissions_2831	struct * IM_PERMISSIONS_2831	
ex_customer_permissi	struct *	

on_2831	EX_CUSTOMER_PERMISSION_2831	

Note:
- Here, ex_customer_permission_2831 acts as an output parameter.

Output Parameters

See Input Parameters' note.

Logic (refer code 13.4.1)

This function executes following steps in the same order:

```
- gets GMT time
- initializes environment for rollback, incase of error.
- retrieve userid from input parameter, im_permissions_2831.
- check if record exist in CUSTOMER table for customer with 'userid' value.
- if record doesn't exist for userid in CUSTOMER table,
        - return error.
- else
        - check no. of records for userid in CUSTOMER_PERMISSIO table
        - if record(s) exist in CUSTOMER_PERMISSIO table
                - if modify_datetime stamp from input parameter is not an empty string of
                20 characters
                        - retrieve modify_datetime for userid from
                        CUSTOMER_PERMISSIO table
                        - if input.modify_datetime < database.modify_datetime,
                                - return error no 4001
                - delete all records for userid from CUSTOMER_PERMISSIO table
        - create record(s) in CUSTOMER_PERMISSIO table using values from input
        parameter, im_permissions_2831
- set output parameter, ex_customer_permission_2831, to GMT time.
- return
```

9.2 read_customer_permissions_p

Function

This function is invoked when a request comes to cixsmainp process, from cis process to retrieve records from the CUSTOMER_PERMISSIO table.

Input Parameters

Parameter	DataType	Remarks
im_customer_permission_2829	struct * IM_CUSTOMER_PERMISSION_2829	
ex_permissions_2829	struct * EX_PERMISSIONS_2829	

Note:
- Here, ex_permissions_2829 acts as an output parameter.

Output Parameters

See Input Parameters' note.

Logic (refer code 13.4.2)

This function executes following steps in the same order:

```
- initializes environment for rollback, incase of error.
- retrieve userid from input parameter, im_permissions_2831.
- check if record exist in CUSTOMER table for customer with 'userid' value.
- if record doesn't exist for userid in CUSTOMER table,
        - return error.
- else
        - retrieve records from CUSTOMER_PERMISSIO table for a given userid.
        - if no records found,
                - return error.
        - else
                - create a linked list of EX_PERMISSIONS_2829 type from the records
                returned from CUSTOMER_PERMISSIO table.
- return
```

10 Source Files

List of files modified/created:

10.1 FCC

File name	Remark	New/Modify
Xaction.pl		Modify
OpcodeToSocket.pl		Modify
cisTestBody.pl		Modify
CRUDcustPerm.pl		New

10.2 FCD

File name	Remark	New/Modify
CRUDcustPerm.c		New
Makefile.xaction.s700.10		Modify
services.dat		Modify
services2.dat		Modify

10.3 CISapi

File name	Remark	New/Modify
CRUDcustPerm.c		New
Makefile.api.s700.10		Modify

10.4 CIDB

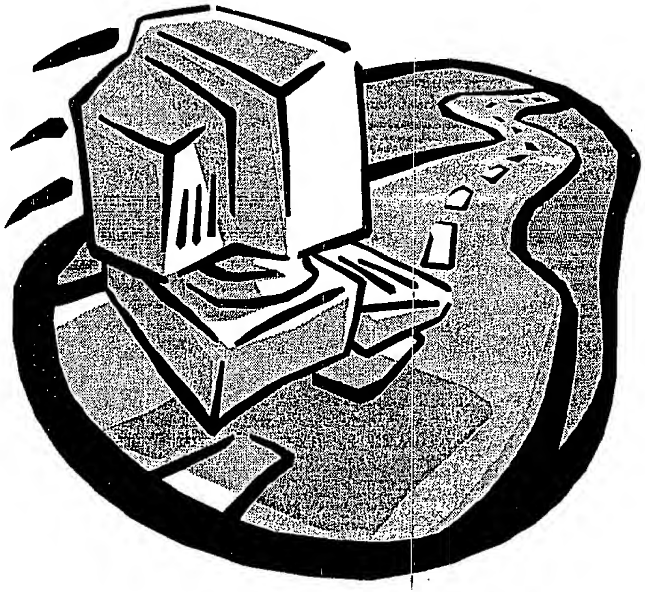
File name	Remark	New/Modify
cixsprad.c		Modify
dbapi_aux.ec		Modify
dbapi.ec		Modify

11 DataFlow

EXHIBIT B

IT Resource Center

ITRC 2.0 Release MR Meeting



[redacted]

[redacted]

invent

HP Confidential

[redacted]

Purpose and Process for MR Meeting

Resource Center

- Purpose of Meeting
  - to obtain agreement from Program Office and Core Release Team members to deploy the ITRC 2.0 version.
- Process - review the following:
  - Release Scope (from Release Commit)
  - Planned vs. Actual
  - Performance/Stress Testing Results
  - Training Conducted
  - System Test Data
  - Defect Tracking
  - Deployment Planning
  - Recommendations to Deploy
  - Open Issues for Post Deployment

Release Scope (from Release Commit Presentation)

IT Resource Center

- Improve Performance and Availability through “refresh” of existing infrastructure.
- Improve ability to obtain accurate information more quickly through improvements to search capability and navigation as well as through the development of on -line collaboration tool.
- Release of e-commerce capability in Europe (with changes for US/Canada).
- Deployment of HWCM in Luxembourg and Belgium as well as localization of this feature.
- Removal of Y2K tool references.

Planned vs. Actual

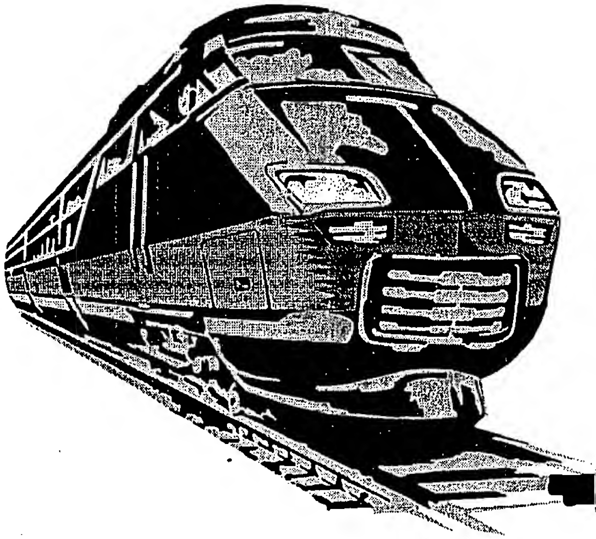
IT Resource Center

Planned Features

- Planner Navigation
- Search Assistance
- HUB Localization
- HWCM Localization
- Self Solve Enhancements
- Collaboration Pilot
- Y2K Product Removal
- HWCM Deployment Bel/Lux
- SWCM Fastcgi
- E-commerce Maintainer
- E-commerce Trainer
- HUB Productivity Tool
- Optimize CIDB
- WW CIDB
- System Handle Refresh
- EIA - Post Wave 1
- Shared Services - SSA
- Planner EGOR
- Rapid Login

Actual (All Aboard!)

- Planner Navigation
- Self Solve Enhancements
- Y2K Product Removal
- HWCM Deployment Bel/Lux
- SWCM Fastcgi
- E-commerce Maintainer
- E-commerce Trainer
- Optimize CIDB
- System Handle Refresh
- Planner EGOR
- Rapid Login



Planned vs. Actual (continued)

IT Resource Center

Change Requests

- CR #62: Remove HP Optimize
- CR #64: Changes to optimize Email
- CR #65: Remove "Call me" from HWCM
- CR #68: RCE Recognition
- CR #69: Granular Suppression
- CR #70: SUM - Fast CGI
- CR #71: Results List by 10 – CKI
- CR #71: Results List by 10 – EMSE
- CR #72: Custom Patch Manager
- CR #74: "Spider robots"

Enhancements

- User Class Topic
- A-box Filter
- Enhanced Search Panel
- TDK Feedback Mechanism
- Metrics Collection
- Localization – E-Commerce
- Localization - SUM
- Localization - SWCM
- Localization - Patch Database
- Localization - "Usage" page
- H2SC

Performance (stress testing)

IT Resource Center

Rapid Login - Pass

- Significantly decreases response times of users with 2 or more system handles
- Current implementation strategy to set Pivot Point to 1 to optimize performance for all users at all system handle levels

CKI - Pass

- Search response times have noticeably reduced based on a decrease in time spent at the repository level

EMSE - Pass

- Response times have not increased. They have decreased slightly due to CKI performance enhancements

E-Commerce - Not Complete

- Lack of an accurate system test environment has limited testing due to firewall constraints
- Testing against Pdapi engine indicates that 10 simultaneous users does not introduce performance issues

SUM - Pass

- Results indicate that fast-cgi has no apparent impact on SUM response times at all user levels

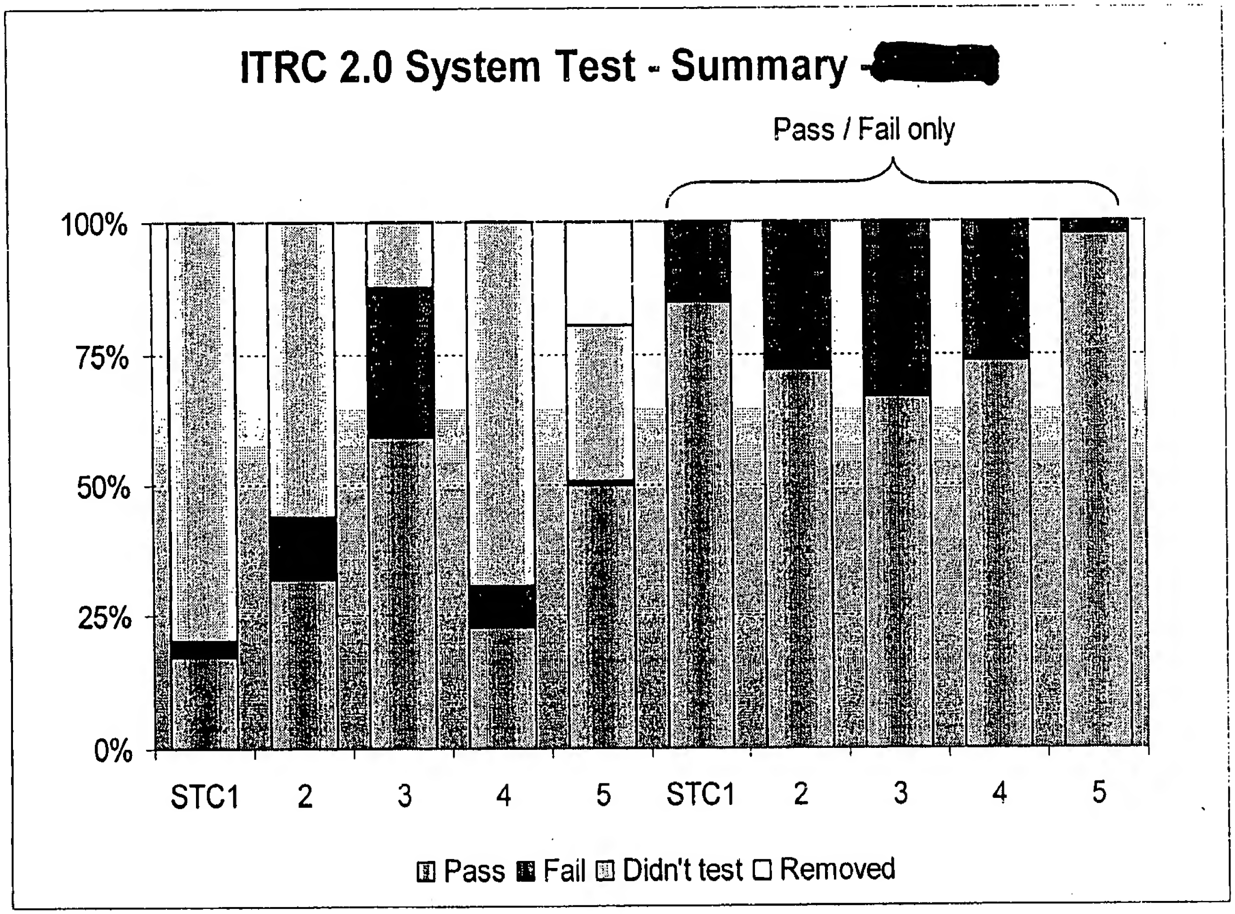
Training Conducted

IT Resource Center

- HUB (MyITRC Project)
- Remove HP Optimize from ITRC (#62)
- Optimize Email Routing in Assistant (#64)
- Pronto
- E-Commerce (Maintainer)
- System Handle Refresh
- Optimized CIDB
- Rapid Login
- WW CIDB
- Y2K Removal
- CPM Changes to ITRC 2.0
- EGOR

System Test Data

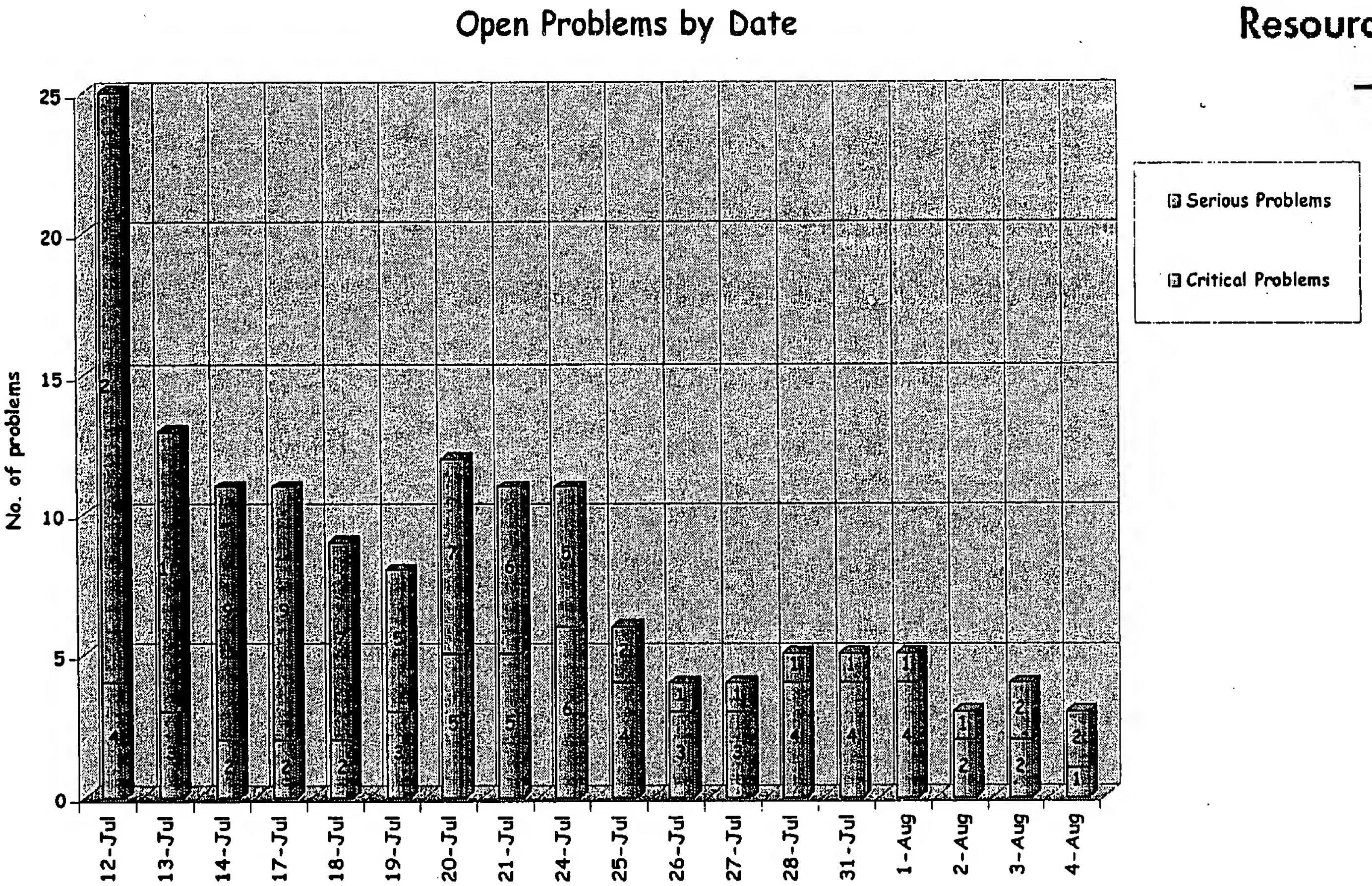
Resource Center



ITRC 2.0 System Test - Summary - [redacted]

Pass / Fail only

Defect Tracking *



*Number of system test defects logged in TBUG and Problem Reporting Log. This number does not include CHART or other non-ASL Product Support tracking systems.

Deployment Planning

IT Resource Center

Status of Deployment Plan: *Plan Complete*

Open issues from Dry Run:

GBIT-E	"CPM needs to be fully tested."
GBIT-AM	"Would be more confident in the future if the dry run environment was taken back to match current production environment rather than installing the same version over the same version."

GBITs MR Recommendation:

Resource Center

"Worldwide, the recommendation is to proceed with the ITRC 2.0 release."

invent

Product Support
MR Recommendation:

Resource Center

GO !

invent

[redacted]

Test Management Team
MR Recommendation:

IT Resource Center

GO !

Program Office
MR Recommendation:

IT Resource Center

GO !

Open Issues for Post Deployment

IT Resource Center

- TBug 0360 - incorrect URL in an error message
- SWPI 24 - Problem with Permissions in E-commerce
- Dry Run
  - userid and username are not displayed from browser to browser for Forums and Planner
  - In leftbar, cannot select Planner then go to Trainer.
- Setting mail alias for CIDBCONN
- TBug ? - Missing "Refund" button on localized E-commerce pages.

EXHIBIT C

Test Plan for RapidLogin (CIS stack) E.2.0

CONTENTS

REVISION HISTORY

Date	Revision Number	Changes	Author
[redacted]	1.0	Original Version. Created template for the Test Plan	Gautam Shah

References

a) CIS FCC/FCD tutorial

Rapid Login - Introduction to the Application

To reduce worst case login time by eliminating the need to completely rebuild permissions during each login.

Today, permissions are dynamically built for each entity that a user has linked. The greater the number of entities (system handles, warranties, and e-orders) the longer the permissions build takes. In addition, the greater the number of potential permissions (as defined in permission rules), the longer it takes to build permissions for each system handle, warranty, or e-order. Permission build time increases linearly based on the number of entities. Similarly, Permission build time changes linearly as the number of Permission Rules changes.

build_time = some_constant * number_of_permission_rules * number_of_entites

To eliminate the need to build permissions at each login for a user, permissions for that user can be stored in CIDB. The complication with this solution is maintaining the stored permissions in an accurate state, which means that the ITRC must know when stored permissions are outdated and should then rebuild them. At these times the user will need to wait as the permissions are adjusted. The access time from CIDB is expected to be a constant.

Affected Areas:

The service layer needs to know when to rebuild or adjust a user's permissions. This occurs when the status of a user's collection of entities changes. Status can change when the following happens:

- A user claims a new entity. When this happens the service layer should adjust permissions for that user by adding permissions for the new entity
- Re-assign:
  A user re-assigns an entity to another user. When this happens the service layer should adjust permissions for the entity for that user by removing the entity's permisssions.
  A user is re-assigned an entity from another user. When this happens the service layer will need to know to adjust permissions for the entity for the new owner the next time he or she logs in.
- Entity sharing with other users. When sharing changes the service layer needs to know to adjust the permissions for the new secondary user of the entity the next time he/she logs in.
- A change in permission rules means that permissions for all users should be rebuilt. The service layer will need visibility to the fact that a particular ITRC release includs a permission rule change so that permissions are updated the first time users login after the release.
- A permision is expended When an e-order item is expended, permissions are adjusted and put to CIDB.
- An entity changes state A background process notices that an entity changes and sets the dirty flag. Entities can expire, unexpire or have other attributes change. See Concerns regarding entity state.
- User Registration If user permissions are stored, putpermissions is called. In the current design, this does nothing.
- Modify Registration If the user modifies his or her registration, all of the entities must be rebuilt.

Special permissions are associated with a user. It is expected that user permissions object will essentially remain as is.

Benefits

Reduced login time except immediately following an ITRC release that includes a permission rule change.

Test Plan

This test plan explains different test cases for RapidLogin. It also describes the interface specifications between CIS FCD/FCC layer and service layer.

The scope of the test cases, described here, is limited to testing CIS stack using testbench only. The events generated through various operations at service layer, are out of the scope of this test plan.

There are 4 basic operations supported for CUSTOMER_PERMISSIO table, used by RapidLogin.

- Create
- Read (GetPermissions)
- Delete
- Update (PutPermissions).

The test cases for each of these 4 operations and their variations are described, separately, in following

EXHIBIT D

☒ [Rapid Login Banner]

Information Technology Resource Center

Performance Improvement for ITRC Login

Persistent Storage of Permissions

For Internal Use Only

Version 1.2

[redacted]

Authors:

Robert Hull (Contractor)
Gautam Shah

Overview

Purpose:

To reduce worst case login time by eliminating the need to completely rebuild permissions during each login.

Today, permissions are dynamically built for each entity that a user has linked. The greater the number of entities (system handles, warranties, and e-orders) the longer the permissions build takes. In addition, the greater the number of potential permissions (as defined in permission rules), the longer it takes to build permissions for each system handle, warranty, or e-order. Permission build time increases linearly based on the number of entities. Similarly, Permission build time changes linearly as the number of Permission Rules changes.

```
build_time = some_constant * number_of_permission_rules * number_of_entites
```

To eliminate the need to build permissions at each login for a user, permissions for that user can be stored in CIDB. The complication with this solution is maintaining the stored permissions in an accurate state, which means that the ITRC must know when stored permissions are outdated and should then rebuild them. At these times the user will need to wait as the permissions are adjusted. The access time from CIDB is expected to be a constant.

Affected Areas:

The service layer needs to know when to rebuild or adjust a user's permissions. This occurs when the status of a user's collection of entities changes. Status can change when the following happens:

- *A user claims a new entity.* When this happens the service layer should adjust permissions for that user by adding permissions for the new entity.
- Re-assign:
    - *A user re-assigns an entity to another user.* When this happens the service layer should adjust permissions for the entity for that user by removing the entity's permisssions.
    - *A user is re-assigned an entity from another user.* When this happens the service layer will need to know to adjust permissions for the entity for the new owner the next time he or she logs in.
- *Entity sharing with other users.* When sharing changes the service layer needs to know to adjust the permissions for the new secondary user of the entity the next time he/she logs in.
- *A change in permission rules* means that permissions for all users should be rebuilt. The service layer will need visibility to the fact that a particular ITRC release includs a permission rule change so that permissions are updated the first time users login after the release.
- *A permision is expended* When an e-order item is expended, permissions are adjusted and put to CIDB.
- *An entity changes state* A background process notices that an entity changes and sets the dirty flag. Entities can expire, unexpire or have other attributes change. See Concerns regarding entity state.
- *User Registration* If user permissions are stored, putpermissions is called. In the current design, this does nothing.
- *Modify Registration* If the user modifies his or her registration, all of the entities must be rebuilt.

Special permissions are associated with a user. It is expected that user permissions object will essentially remain as is.

Benefits

Reduced login time except immediately following an ITRC release that includes a permission rule change.

Proposed Changes to ITRC

FCC:

- **Manage Permissions stored in CIDB via CISFCC**

  Modify the Services layer (FCC) to Get and Put Permissions via the CISFCD. Following is a list of areas known to require modification for the CISFCD permission calls.

    - Get
        - Login (SWOP, WWW, Mail, Batch) - GetPermissions. This is done during

BuildPermissions. Do not Get if less than ten entities. See assumptions.

- Put Always check modify-datetime stamp. If expired, ask user to logoff. The datetime stamp is stored as part of the permissions structure.
  All the following cases occur during either BuildPermissions or AdjustPermissions.
  - Login (SWOP, WWW, Mail, Batch), for dirty entities AdjustPermissions then call PutPermissions.
  - Register User, does a buildPermissions. The unlikely chance of a race condition occurring is ignored.
  - Expend E-Order items and Claim Entity (System Handle, Warranty and E-Order) implicitly call AdjustPermissions. Add call to PutPermissions.
  - Entity Reassign- When an entity is reassigned, implicitly call AdjustPermisions. This implicitly calls PutPermissions.
  - Modify User Information-Does a build permissions and a PutPermissions.
  - Background detection of Dirty bit. Calls GetPermissions then does an AdjustPermissions followed by PutPermissions.

Multiple Instances
A check for a race condition is necessary. If multiple instances of the same user add different entities. The copies of the permissions in session would be different. This could be handled by saving the modify date/timestanp and checking at put. If this fails, a forceful write occurs to the dirty buffer marking the entity needs to be replaced in permissions. The user is then navigated to the login screen. If the failure occurs during login, the GetUserProfile and BuildPermissions is attempted three times.

A similar problem occurs regarding user info. If the user info modify datetime stamp is stale, the user is navigated to the login screen. The user's dirty buffer is marked setall. Admitedly user hostile, but the safest and expedient solution to an unlikely problem.

Behavior of modify_datetime Stamp
The modify_datetime stamp allows the FCC to determine if a race condition has occurred regarding the "put" of permissions. Permissions in session are instantiated by: building permissions, "get"ting of permissions and adjust of permissions. `cis'PutPermissions` is called after an adjust of permissions. It is TBD what should be done if `_adjustPermissions` fails. If the adjust was done on a permission set from a `cis'GetPermissions`, then the modify_datetime stamp should match. If not the user is logged out by FCC. However, if the permissions are from a complete `BuildPermissions`, then the modify_datetime stamp is marked with twenty blanks meaning a new build. The check of timestamp is ignored.
timestamp actions

- GetPermissions - return datetimestamp of retrieved permissions.
- PutPermissions - return datetimestamp of newly stored permissions.
- PutPermissions - pass datetimestamp of recently adjusted permissions. This stamp is either the datetime stamp of the permissions that current instance is built on OR a special flag indicating completely rebuilt.
- PutPermissions - returns an error if the timestamp doesn't match the copy in CIDB and the stamp is not the special flag. If this occurs the FCC logs out the user. The FCD returns MCI_ERR_NUM_OLD_DATA, 4001, for this error.

- **Call Set and Clear Dirty Permissions Buffer**

A preference buffer indicates when permissions need to be rebuilt. Flags can be set on a per entity basis or for all entities associated with a UserID. The per entity flag indicates whether to rebuild or remove permissions associated with the entity.

Following is a list of Services layer areas to be modified to set, setall, and clear the dirty permissions flags.

- Set

        - Entity Reassign - Set the Entity's dirty flag for the target UserID
        - Entity Share (Add, Adjust Remove) - Set the Entity's dirty flag for the secondary UserID
        - Entity Expire - Set the Entity's dirty flag when an entity expires
        - Entity Delete - When an entity is deleted a background process, set the dirty bit for the former entity owner
        - Deletion of a User that owns an entity shared with secondary users
    - Setall entities for a Userid
        - Permission Rule changes - When permissionRules change, the install script must set the setall dirty flag for every user.
        - Modify User - set the dirty buffer for user to Setall. Otherwise, build permissions does a get only.
    - Get and Clear all dirty flags associated with a user.
        - Login, After PutPermissions is completed dirty flag is cleared.
        - After, background process automatically updates permissions, clear dirty Buffer.

- **Manage Dirty Buffer: Set, SetAll, GetClear Permissions Flags**

  Dirty flags use a buffer stored and retrieved through a Customer Preference Buffers.

  The buffer contents maybe empty, a string "ALL", or a string of entity mode pairs. The elements are joined by ':'. If the buffer is empty, the stored preferences are current. If "ALL", the preference mustbe completely rebuilt. If a string of entities, the entities identify the entities to be adjusted with the attached mode.

    - **SetDirty**
      Input
        - UserID
        - EntityID
        - Mode

      Retrieve current buffer for UserID. If the buffer is "ALL" return. If the buffer is empty, set to EntityID and Mode. If the buffer exists, search for EntityID in list. If in list return if modes are equal, otherwise set mode to replace. If not in list append to list. Write current buffer for UserID.

    - **SetAllDirty**
      Input
        - UserID

      Write preference buffer with contents "All" for UserID.

    - **GetAndClearDirty**
      Input
        - UserID

      Output an array of
        - EntityID
        - Mode

      Retrieve current buffer for UserID. Write preference buffer with contents undef for UserID. Return retrieved buffer.

- **Session: Permissions**

  No changes to Session Permissions functions are anticipated. Note, whenever SessionPermissions are updated, the CIDB copy of Permissions should be updated.

o CISFCC

Changes to the CISFCC follow:

- Add calls to GetPermission and PutPermission similar to CISFCD calls.
- Enchance cisTestBench5.pl and cisTestBody.pl to call the new CISFCC calls for testing and administration.

o SuperUser Utility

Add the ability to force a build permissions and store to CIDB on a given user from the SuperUser utility. This occurs regardless of the number of entities claimed by the user. This build would clear the dirty buffer. This is part of the modify user capability.
It maybe a future to program delete user.
It maybe a future to program unlink system handle.
It is a future to program modify system handle.

It is unclear how to know when to manually rebuild permissions. Current thought is when the user either complains or we notice a timeout error for login. See Assumptions or Concerns.

o Customer Information Lookup and SuperUser Utility

Both Customer Information Lookup and SuperUser Utility would like a feature to view permissions stored to cidb. This could include viewing permissions stored in CIDB as well as viewing dirty buffers.

o Background Processes

Following is a list of background or batch processes for this change.

- Background script to set dirty flag if entity changes. At midnight, launch a script to check each entity for expiration. If so, mark the associated users dirty buffer for the entity. If the entity is a system handle, check the modify-datetime stamp. If this is newer than the permissions modify-datetime stamp, set the dirty flag for the owning user. This assumes the "System Handle Refresh" changes for 1.5 set the modify-datetime stamp. If this is a warranty, check the modify-datetime stamp and the expiration datetime. If an e-order, check the modify-datetime stamp and the expiration datetime.
- Batch script to set dirty bits when PermissionsRules change. During install of a new release with new PermissionRules, mark all userids to rebuild all entity permissions.
- Background to Adjust and Put Permissions on users. This script could be postponed. However, it will be difficult to initialize users with a large number of entities without this. This script should find and adjust UserIDs with the most dirty entities first. If the modify datetime stamp is stale, restart on that user. It may be appropriate to have multiple adjust background processes. If so a pool of UserIDs to adjust should be maintained to prevent conflicts such as two scripts attempting to adjust the same user. This condition could occur if the process takes a long time to adjust all users. For example if the script starts each day and it takes more than one day to adjust all users.

FCD:

o Get and Put System Permissions

Following FCD operations are proposed to handle new CIDB schema

a) GetPermissions

This function reads derived permissions for a given customer from CIDB database.

Required parameters are:
TOBJ string UserId

Returns:
TOBJ permission structure (systems-perms-info)
modify_datetime timestamp for DB change

Errors:
If UserId doesn't exist or not specified, returns error MCI_ERR_NUM_USER_NOT_FOUND (1002).

b) PutPermissions

This function updates derived permission in CIDB for a given customer. If the modify_datetime is wrong, the write is aborted and an error raised. If the modify_datetime is marked build twenty blanks, then do not check. See modify_datetime for more details.

Required parameters are:
TOBJ string UserId
systems-perms-info Tobj Permission structure
modify_datetime timestamp from previous getPermissions

Returns:
modify_datetime timestamp for DB update

Errors:
If UserId doesn't exist or not specified, returns error MCI_ERR_NUM_USER_NOT_FOUND (1002).
If modify_datetime differs, return an error, MCI_ERR_NUM_OLD_DATA (4001). Do not write passed in permissions to CIDB.

CIDB:

At present, permissions are calculated (build) at login time using data from user's profile and preference information. For some users this exceeds the five minute login timeout. The permissions includes free permissions, derived permissions and special permissions.

- Free permissions are read from PermissionRules.pl file at login time.
- Derived permissions are calculated online from user profile information.
- Special permissions are read from customer preference buffer.

Free permissions and special permissions will be calculated and stored using existing methods. However, derived permissions will be stored in CIDB. The proposed change is to modify/extend CIDB schema by adding a new table *Customer_Permission*, to store customer 'permissions' for faster access. The sequence number is used to create a row. However, retrieval does not use the sequence number. Retrieval is all rows for a UserID. The proposed format for this table is as below:

Customer_Permission

Attribute	Type	Constraint
UserId	Char(30)	NOT NULL
EntityId	Char(??)	NOT NULL
Sequence Number	Int	
Permissions	Char(??)	NOT NULL

The 'Permissions' buffer lists all the permissions for a given unique pair of UserId and EntityId. More than one permissions are separated by ':'. (See Assumptions or Concerns for more information).

Following is an example of how actual data will be stored in this table.

UserId	EntityId	SeqNum	Permissions
CA1	W&C-CLS	0	ENT_GEOG:ECOMMERCE_AVAILABLE
CA1	CA1_64	0	SW_DEL:SW_LIST
CA1	CA1_72	0	HWCM_SUMRY:HWCM_DETLS:HWCM_SUBMIT:
CA1	CA1_72	1	HWCM_SPCL_USR
CA2	CA1_83	0	SW_SUBMIT:SW_PS_LTD:SW_APP_LTD:SW_SUBMIT_P:SW_GET_P

Performance expectations

The performance for users with few entities (maybe ten), should slow marginally. The performance for users with over fifty entities should improve. Below are the order of magnitude calculations for some operations.

Login Time

Number of Entities	As Is	Best	Worst	Likely
Formula	C1 + C2 * N	C1 + C3 + C4 * N + C5	C1 + C2 * N + C3 + C4 * N + C5	C1 + C2 * .1 * N + C3 + C4 * N + C5
0	0.5	2.5	2.5	2.5
10	5.5	3	8	3.5
25	13	3.75	14.25	4.75
200	100.5	7.5	107.5	36.5

Note: The current release has an extraneous call to getSpecialPermissions for each entity. This should be obsolete. However, until then the current "AsIs" takes an extra call to the FCD and CIDB. This adds an extra .5 seconds per entity with these assumptions.
C1 = .5 seconds - time to build User Permissions and overhead
C2 = .5 seconds - time to build Permissions for an Entity
C3 = 1 second - time for an FCC FCD transaction (likely .5)
C4 = .05 seconds - time for CIDB access from FCD (likely slower)
C5 = 1 second - time to read and clear Dirty buffer
Assume 1/10 of N changed entities for likely login.

Other actions will take longer after the change.

Adjust Users' Permissions

As is takes DeltaN * C2 to complete. Add C3 + C4 * N to store permissions.

Dirty Another Users' Permissions

Add C5 to read and write dirty buffer.

Rebuild Users' Permissions

As is takes C1 + N * C2 to complete. Add C3 + C4 * N + C5 to store permissions.

Assumptions or Concerns

- If a user's dirty flags are set while she or he is logged in, it is OK if the user continues with stale permissions.
- The setall dirty flag is better than getting all entities and setting each entity individually.
- When a user's actions cause an AdjustPermissions, it is appropriate to retrieve the dirty buffer and AdjustPermissions on the dirty entities.
- Special and User permissions will be built as is.
- Customer_Permission table
  - combination of UserId, EntityId and Sequence Number is a unique combination.
  - Colon (':') is a separator between permissions. Colon is not allowed as a valid character in EntityId.
- If Delete User is implemented, the impact on permissions will have to be coded.
- It is unclear how to detect all entity changes. Expiration can be detected. Unexpire is not as obvious.
- Rebuilding permissions for less than ten entities is quicker than retrieving the save permissions.
- The new System Handle refresh design assures us the entity modify_datetime stamp would be valid indicator of whether an entity should be rebuilt. The background check will check an entities modify_datetime against the permissions modify_datetime. If the entity is newer, the entity is marked dirty.
- It is unclear how to know when to manually rebuild permissions. Current thought is when the user either complains or we notice a timeout error for login.

Alternatives or Tradeoffs

- Dirty Flags could be implemented as a new feature for CIDB or by using the existing customer preference buffer schema. The current schema limits the buffer to 10,000 bytes. If an entity takes on average 30 bytes to be uniquely identified, we have room for a total of 333 dirty entities. I expect this to be more than adequate. If we exceed the buffer size, the program could mark the user as rebuild all preferences. The preference buffer schema should quicker to develop.
- The expiration of an entity can be managed by either a background script or by storing the expiration timestamp with permissions. The time to check all entity-user pairs could be large. It was felt to be more expedient to store the expire stamp with the permissions and check the timestamp at retrieval. This could be done similar to the time-bomb check for special permissions.
- The dirty flags were briefly designed with the entity buffers (eorder_buffer, sys_handle_buffer and warranty_buffer). However, it was recognized that this would be difficult for entity assignment or entity sharing.
- Race conditions could be dynamicly updated. However, it was recognized that this could be an infinite loop. Also, the fall back of logging out the user still had to be coded. It was noted that the race scenario is unlikely, however, it must be coded for. The logout was felt appropriate. If the race occurs with a background process, the background process restarts processing on the user. If that also fails, the process restarts on all users.
- The CIDB row size is about 1000 bytes. This is smaller than if an entity has all permissions concatenated. If the permission string exceeds the row size, the permission string will be stored in a second row. If the buffer is needed regularly, access time is doubled for get and put. An alternative was considered to store one permission per row. A concern with many cidb accesses for all rows was raised.
- The dirty flags are stored in a customer preference buffer. It was considered to store dirty flags in the entity preference buffer. However, this is problematic when the entity is deleted or shared.
- The storing of User level permissions is an incremental change for later.

Testing and Benchmarks

The goal of this change is to improve the performance during login. By testing the performance before and after we can confirm our conjecture that these changes will improve login performance. Similarly, changes elsewhere should be verified for any unexpected performance changes. Testing should check the time (elapsed and compute) for the following actions:

- Login of a user with 100 handles.
- Login of a user with a combination of 150 entities.
- Login of a user with no entities.
- Login of a user with one entity.
- Login of a user with only shared entities.
- Login of a user with an expired entity.
- Login of a user with a deleted entity.

The following cases should be tested for correct behavior. It is not necessary to benchmark performance.

- Login of a user with 30 e-order entities.
- Expend an e-order.
- Purchase an e-order.
- Claim a system handle.
- Claim a warranty.
- Reassign an entity.
- Share an entity.
- Change the sharing of an entity.
- Delete the sharing of an entity.
- Login of a user after a PermissionRule change. (Try with several entities)
- Change the special permissions of a user.
- Change the free permissions.

Task Estimate

- Benchmark existing - 1/2 week
- Code - 37 person days or 3 1/2 weeks with 2 programmers.
    - Add Get and Put Permissions within FCC.
        - Add GetPermission call 1/2 day
        - Add PutPermission call 2 1/2 day
    - Add Set and Clear Dirty Permissions within FCC.
        - Add call setDirty 1 day
        - Add call getAndClearDirty 1 day
    - Add Dirty Permissions management within FCC.
        - Add setDirty 1 day
        - Add setAllDirty 1 day
        - Add getAndClearDirty 1 day
    - Modify CISFCC (Get and Put Permissions).
        - fccGetPermission 1/2 day
        - fccPutPermission 1/2 day
        - cisTestBench 1 day
    - SuperUser Utility
        - add set user permissions 3 days
    - Install script.
        - modify install script 2 days
    - Entity change background script.
        - entity change script 5 days
    - Update User background program. (postpone?)
        - Update user script 5 days

  - Modify CISFCD (Get and Put Permissions).
    - fcdGetPermission 1 day
    - fcdPutPermission 1 day
  - Create, Read, Update and Delete CIDB rows.
    - fcdCreateCIDBPermission 1 day
    - fcdReadCIDBPermission 2 days
    - fcdUpdateCIDBPermission 2 days
    - fcdDeleteCIDBPermission 1 day
  - Build Permissions Rows within FCD.
    - CIDB update 2 days
- Testing and final benchmark - 2 and 1/2 weeks
- Document FCC 3 days
- Document FCD 3 days
- Buffer - 1 week

☒ [Rapid Login Footer]